AFRL-IF-RS-TR-2006-12
Final Technical Report
January 2006

DYNAMIC COURSE OF ACTION DECISION TOOL

BAE Systems

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2006-12 has been reviewed and is approved for publication

APPROVED: /s/

JOSEPH A. CAROLI
Project Engineer

FOR THE DIRECTOR: /s/

JAMES W. CUSACK
Chief, Information Systems Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	JANUARY 2006	Final Aug 2002 – Aug 2005

4. TITLE AND SUBTITLE
DYNAMIC COURSE OF ACTION DECISION TOOL

5. FUNDING NUMBERS
C - F30602-02-C-0201
PE - 62702F
PR - EBO0
TA - 00
WU - 11

6. AUTHOR(S)

Mr William Sexton III, Jonathan P. Stucklen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
BAE Systems
Advanced Information Technologies, Inc.
6 New England Executive Park
Burlington Massachusetts 01803-5012

8. PERFORMING ORGANIZATION REPORT NUMBER

TR-1690

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL/IFSA
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2006-12

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Joseph A. Caroli/IFSA/(315) 330-4205/ Joseph.Caroli@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*
This report summarizes the results of an effort aimed at developing a Predictive BattleSpace Awareness, PBA, capability for Time Sensitive Targeting. The DCOAD, Dynamic Course of Action Decision, tool integrates the PBA process with the air battle plan to predict the likelihood of Time Sensitive Target occurrence, the likelihood of their discovery, the likelihood that once discovered, they can be successfully attacked, and the overall probability that TSTs can be successfully countered given the configuration of the daily air battle plan. DCOAD is a tool that generates prediction and presents them in a form of probabilistic maps, indicating the likelihood of TST occurrence in a given geographical area and the strike and ISR coverage in those areas. These maps are overlaid on a situational display to provide the operators with information regarding "gaps" in TST coverage. The report includes information on the DCOAD predictive models, software architecture, system architecture, and the operational utility of the tool.

14. SUBJECT TERMS
Time Sensitive Targets

15. NUMBER OF PAGES
48

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Appendixes

List of Figures

List of Tables

1. Executive Summary

The primary objective of this program was to develop a Dynamic Course of Action Decision (DCOAD) tool, a software system to aid the Plans and Combat Operations Divisions of an Air Operations Center (AOC) to evaluate time sensitive targeting operations in the context of an effects-based plan. Further, the DCOAD software team attempted to develop the first practical example of a predictive battlespace awareness (PBA) tool that can serve operators at both the operational and tactical level of warfare.

To understand why DCOAD is a PBA tool, you need to understand what PBA is. PBA attempts to anticipate our adversary's next move before he makes it. It includes four primary pillars:

- Intelligence Preparation of the Battlespace
- Target Systems Analysis
- ISR Strategy and Planning
- ISR Employment and Assessment

PBA is especially important in Time Sensitive Targeting operations because of the fleeting nature of the targets, limited asset availability, compressed decision timelines, and strike asset responsiveness.

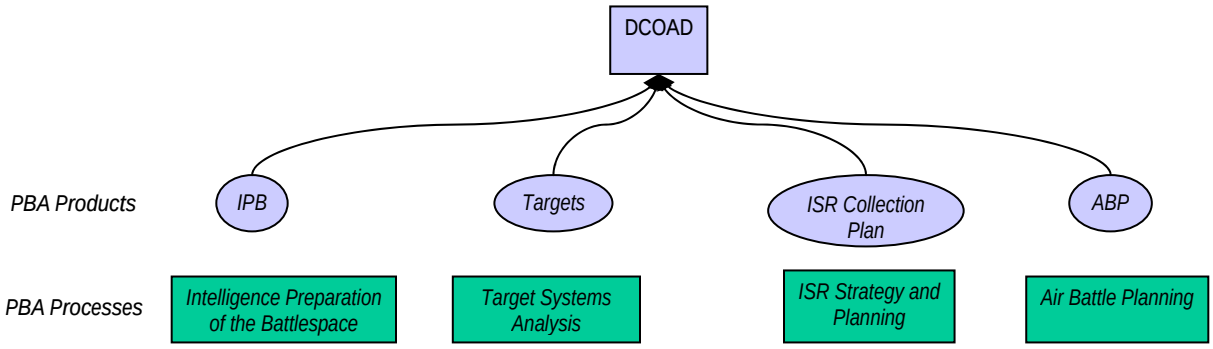

Figure 1 - DCOAD brings together all the products of the PBA process

As shown in figure 1, DCOAD is the first application that brings all the products of the PBA process (plus the daily air battle plan) to help operators predict and visualize:

- the likelihood of TST occurrence
- the likelihood of TST discovery
- the likelihood that, once discovered, TSTs can be successfully attacked, and
- the overall probability that TSTs can be successfully countered given the configuration of the daily air battle plan.

DCOAD's unique probabilistic overlays provide operators a practical, easy-to-interpret display that can be used to assess if the daily air battle plan is adequate to counter anticipated TSTs that might appear in the battlespace. DCOAD generates predictions and presents them in the form of probabilistic maps, indicating the likelihood of TST occurrence in a given geographical area and the Strike and Intelligence, Surveillance and Reconnaissance (ISR) coverage in those areas. These maps are overlaid on a situational display to provide operators information regarding "gaps" in TST coverage.

DCOAD's architecture consists of three major components; the graphical user interface, the shared data services, and the estimator framework. These fit into the paradigm of the model-view-controller architecture used in most graphical applications.

During the course of the three year DCOAD project, we conducted research and development in three major areas. First, we developed predictive models for target occurrence, ISR asset coverage over time, Strike asset coverage over time and composite probability models for successful TST prosecution. Second, we developed a plug-and-play estimator framework for calculating discretized estimates and representing the estimates in a manner that can be displayed either pictorially as graphical overlays or output to external sources as xml files. Finally, we developed a user friendly graphical user interface to control and present actionable information to the operator. Overall, our research and development efforts resulted in a user friendly tool that can be used by planners in the AOC to check the ability of the daily air tasking order to counter potential pop-up targets in the battlespace

While this work represented a major step forward in predictive battlespace awareness research, many areas for further research and development still exist. More representative ISR platform and sensor models need to be developed; models that more accurately present footprint geometries and scan patterns associated with real system. DCOAD is designed as an analysis tool for theater level air campaign. However, the basic estimator framework and GUI could be expanded to support the "close-in fight". DCOAD could provide US Army and US Marine Corps planners a tool to visualize and potentially coordinate ISR support for patrol and cordon & search activities in support of Stability and Support Operations.

2. DCOAD Program Objectives

The objective of the DCOAD effort was to develop technology to support Time Sensitive Targeting (TST) within the context of effects-based operations. As originally envisioned, the resulting DCOAD software was to have the capability to support: (a) Predictive Battlespace Awareness (PBA) showing probable TST locations, available Intelligence, Surveillance and Reconnaissance (ISR) assets and available strike assets; (b) prioritization of preplanned and unplanned targets and missions; and (c) assessment of the impact of TST prosecution on overall campaign plans.

More specifically, the technical requirements for the program called for the design, development and demonstration of software that provided:

- A static TST PBA capability which determined and displayed static PBA data for planning purposes, including:
  - TST likelihood of occurrence as a function of location and time
  - Likelihood of TST discovery as a function of ISR asset availability, location and time
  - Likelihood of successful TST prosecution as a function of strike asset availability, location and time
  - A composite view of the overall likelihood of being able to engage TSTs as a function of location and time given TST occurrence and the availability of both ISR and strike assets.
- A dynamic TST PBA capability which determined and displayed dynamically updated PBA data for execution monitoring purposes, including:
  - TST likelihood of occurrence as a function of location and time
  - Likelihood of TST discovery as a function of ISR asset availability, location and time
  - Likelihood of successful TST prosecution as a function of strike asset availability, location and time
  - A composite view of the overall likelihood of being able to engage TSTs as a function of location and time given TST occurrence and the availability of both ISR and strike assets.
- A prioritization capability which dynamically accessed current and planned mission and target priorities based upon the impact those targets and missions have in influencing the effects-based plan objectives.
- A campaign assessment capability that assesses the impact that diverting missions to prosecute TSTs has on the campaign objective.

As noted Section 6 of this report (Problems and Lessons Learned), the first two objects were successfully achieved, while our ability to achieve the final two objectives was restricted due to limitations in the campaign assessment capabilities available necessary to support DCOAD campaign assessment and prioritization capabilities. This is further discussed in section 6.2.

3. DCOAD Development Approach and Concepts Employed

3.1. Spiral Development Concept

DCOAD software development was conducted under a three phase spiral development concept delivering ever increasing capabilities in each phase. During the "Planning" phase we developed the baseline probability estimators, data interfaces and visualization capabilities needed by AOC operators to use DCOAD to support ATO planning. During the "Execution" phase we developed dynamic probability estimators that condition probability estimates with near real time track data; a capability necessary to allow operators to use DCOAD to support the execution of the ATO. The "Assessment" phase was originally planned to incorporate capabilities in DCOAD to allow operators to assess the impact of TST operations on the effects-based plan; however, many of these capabilities were not developed due to limitations in the campaign assessment capabilities explained in section 6.2.

The spiral development process is an iterative set of sub-processes that include: the establishment of performance objectives; design; code, fabricate, and integrate; experiment; test; assess operational utility; make tradeoffs; and deliver. The goal of spiral development is to allow innovation in technology and operational concepts to occur simultaneously and continuously at many levels and across all functional lines. The Spiral development concept was well suited to DCOAD objectives. PBA visualization and effects-based operations are emerging concepts in the United States Air Force requiring the iteration of operational concepts and technology enhancements to achieve results. Overall, eighteen software units were developed over three spirals in the DCOAD program. The functionality introduced in each phase is described in Table 1 below.

Table 1-Unit Functionality by Phase

Phase	Unit Functionality
Planning	
	GUI Backbone
	Map Entities Shared Data Services
	Parametric Data Manager
	A2IPB Interface
	AODB Interface
	ISRM Interface
	Probability Estimators Base Functionality
	TST Occurrence Estimator
	ISR Asset Coverage Estimator
	Strike Asset Coverage Estimator

	Composite TST Prosecution Estimator
Execution	
	TMS Interface
	Dynamic Estimators
Assessment	
	GUI Enhancements
	ISR Estimator Enhancements
	Block Shape Enhancements
	Strike Estimator Enhancements
	SDT Interface Mockup

3.1.1. Planning Phase

The Planning Phase was by far the most comprehensive phase of the development. The objective of the planning phase was to create a baseline DCOAD capability that allowed planners to predict and visualize:

- The TST likelihood of occurrence as a function of location and time
- The likelihood of TST discovery as a function of ISR asset availability, location and time
- The likelihood of successful TST prosecution as a function of strike asset availability, location and time
- Likelihood of being able to Find, Fix, Track, Target and Engage TSTs given TST occurrence and the availability of both ISR and strike assets.

In the Planning Phase BAE-AIT software engineers concentrated on the development of infrastructure to support DCOAD visualization and estimation. In addition, significant effort was expended in the development of a shared data management service and the creation of interfaces to required data sources. Of the 18 software units developed during the course of the DCOAD project, 11 were developed during Phase I.

3.1.2. Execution Phase

During the execution phase, BAE engineer introduced dynamic data in the form of near real time track data into DCOAD probability calculations. Probability estimators were enhanced and modified to accept dynamic data and an interface to the Integrated Command, Control, Communications, Computer, and Intelligence (C4I) System Framework

ICSF 4.5.2.7 Tactical Management System (TMS) for near real time track data. OpenMap visualization schemes were modified to display track history and predict asset routing based on current near time location data and future planned waypoints. This capability provides operators a way to visualize where assets have been and assess where they might go to correct planned flight paths. Probability estimators were modified to update the likelihood of TST occurrence, discovery and successful prosecution based on near real time track data.

3.1.3. Assessment Phase (Enhanced Capabilities)

During the Assessment Phase BAE-AIT software engineers attempted to integrate several enhancements to the DCOAD; all were not successful. The most significant enhancement came with the introduction of terrain elevation data [Digital Mean Elevation Data (DMED)] into ISR asset probability of detection (Pd) calculations. Terrain variance data available thru DMED was used to condition detection probabilities in higher terrain. The use of terrain variation data to condition Pd values for ISR assets avoided the processing burden associated with traditional line of sight calculations while giving operators an indication of the effect of terrain on the ability ISR assets to discover TSTs.

BAE-AIT software engineers also attempted to improve the depiction of ISR asset footprint information, however, the programming method employed proved to be too computationally intensive, slowing processing time and negatively impacting application performance. As a result, the effort was abandoned to concentrate on implementing the above described terrain variance modeling effort.

Engineers also hoped to provide a "dummy" graphical user interface (GUI) to demonstrate how DCOAD could be used to support the assessment of TST operations on campaign objectives; however, program resources were not sufficient to implement this demonstration capability. Further, the program manager decided the value added for implementing the terrain variance modeling for ISR assets was higher than pursuing the demonstration GUI.

3.2. Software Engineering Institute Capability Maturity Model (SEI CMM) Level III Processes

In summer of 2003, the DCOAD program was selected by BAE-AIT's Defense Program Office to transition to a SEI CMM Level III software process. This introduced a significant amount of overhead to the program in the middle of execution; however, several benefits were realized. The establishment of scheduled peer reviews allowed us to catch software problems earlier in the process. Catching problems during reviews save effort and resources that would have been more expensive to fix later in the software development cycle. The introduction of Level III SEI CMM processes in 2003-2004 inspired a reexamination of the DCOAD Program requirements

and scope. As a result of this reexamination, the Assessment Phase of the program was eventually adjusted to achieve more realistic objectives. The introduction of a Configuration Control Board process for DCOAD allowed priorities to be set and documented, and kept software team members on the "same sheet of music".

4. DCOAD Design

4.1. Predictive Models

Over the course of DCOAD development, several predictive models were created to demonstrate the power and flexibility of our estimation framework. The models fell under 4 basic categories; TST occurrence models, ISR coverage models, strike coverage models, and composite models. The exception is the DMED standard deviation model which is explained in more detail below.

4.1.1. TST occurrence model

The TST occurrence model was developed to represent target likelihood of occurrence based on geographic lines and areas defined in Automated Assistance with Intelligence Preparation of the Battlespace (A2IPB). Specifically, A2IPB Infrastructure Areas and Lines of Communication (LOCs) are used to represent potential target occurrence areas. The probability of target occurrence is computed, over space, using target occurrence confidence levels associated with the A2IPB products as explained below. In addition to a confidence level, the A2IPB Infrastructure Areas and LOCs have associated periods of time over which the confidence levels apply. This data is used to provide an estimate of TST occurrence over time by computing a probability in space for each time slice over the period of estimation.

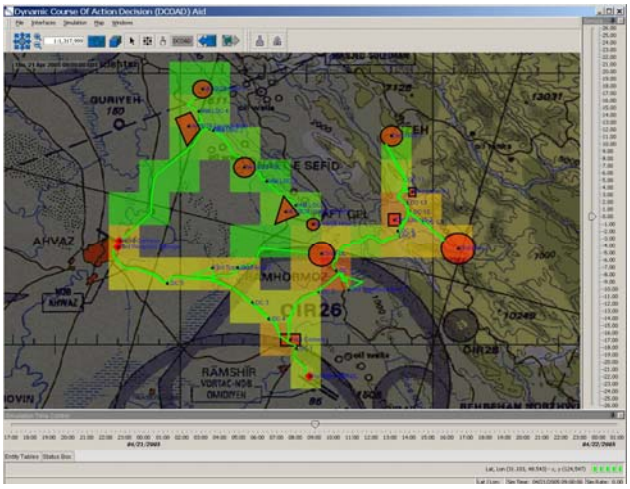

Figure 2- Depiction of the TST occurrence estimate for a region in the Iranian scenario.

A list of equipment types and unit types to be considered TST types are provided to the estimator at initialization time. For purposes of describing the algorithm, this will be called the TST type list. A map from A2IPB Confidence Levels to corresponding probabilistic values is also provided at initialization time. The estimator iterates over all loaded A2IPB Infrastructure Area products. An Infrastructure Area is used to describe a probable area for TST occurrence given the following conditions,: 1) The Infrastructure Area must be designated as an area with probable TST activity; 2) the Infrastructure Area must be a geographical shape (as opposed to a point) and, 3) the Infrastructure Area must be associated with at least one unit for which the unit type is contained in the TST type list. For each unit within the Infrastructure Area, it is determined if the unit is of a type listed in the TST type list. These units are referred to as TST units. The probability of TST occurrence for the Infrastructure Area is then calculated as follows: For each TST unit, the Confidence Level for the association between the Infrastructure Area and the unit is used to obtain a probability of TST occurrence for that TST unit. The final probability of TST occurrence for the Infrastructure Area is the probability of at least one TST unit appearing within the area. Formally, this looks like:

$$P_{area} = \sum_{tstA \in TST\,Units} P_{tstA} \prod_{\substack{tstB \in TST\,Units \\ tstB \neq tstA}} (1 - P_{tstB})$$

$$P_{area} = 1 - \prod_{tst \in TST\,Units} (1 - P_{tst})$$

Where tstA and tstB are units that are considered TST units as defined above and P_{tstA} and P_{tstB} are the probabilities of the TST unit appearing within the Infrastructure Area. P_{area} is the final probability of TST occurrence for the Infrastructure Area. A similar algorithm is used for LOCs to calculate a probability of TST occurrence along LOCs.

4.1.2. ISR asset coverage model

The ISR asset coverage model is used to determine where and when sensors will be available, and how well the sensors are likely to perform. It also considers the impact of retasking in response to TST occurrences. There are three components to the ISR collection model. The first component analyzes how an ISR asset can be retasked from its original collection plan, the second component determines sensor coverage given the asset's position, and the third component determines sensor performance.

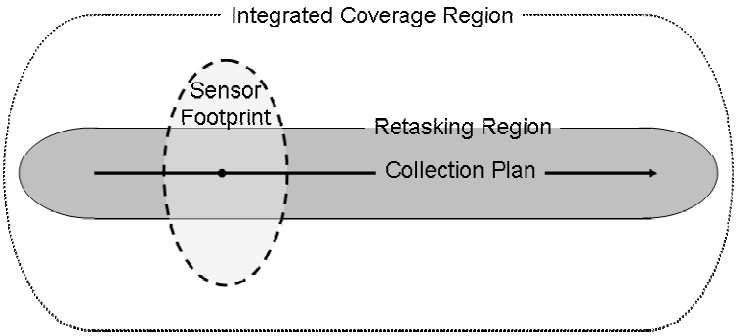

Figure 3 - ISR Asset Coverage Model shows operators where and when sensors will be available

As illustrated in Figure 3, the collection model is based on the original collection plan. Retasking for TST "pop-ups" is typically limited due to the accelerated time scale associated with TST activity, which results in a relatively small retasking region, in the vicinity of the original collection plan. Sensor coverage establishes a region on the ground, sometimes called the "footprint," that is visible to a particular sensor for a given asset position. Sensor performance (e.g., probability of detection or image resolution) is defined for all points within the footprint of the sensor. For simplicity, a circular footprint is used for all ISR assets. Sensor performance includes the effect of terrain through the use of Digital Mean Elevation Data (DMED). The output of the ISR collection model is a map indicating sensor performance integrated over the original collection plan and retasking region.

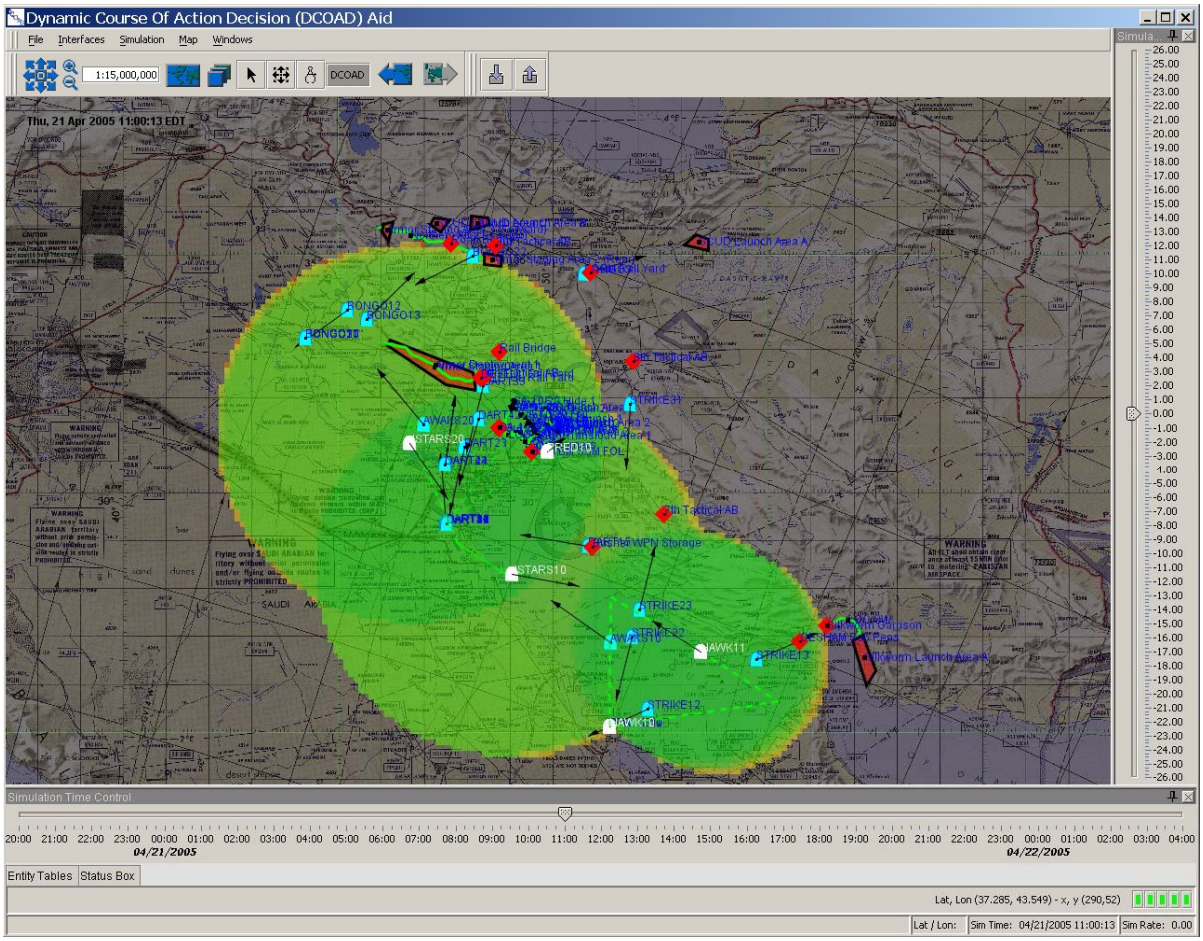

Figure 4 - Depiction of the ISR coverage estimate for the entire Iranian scenario.

4.1.2.1. ISR asset coverage model with DMED

The enhanced ISR Estimator with DMED masking is a modification of the standard ISR coverage estimator that takes into account terrain masking when computing the ISR coverage estimate. This estimator uses a novel approach that employs Digital Mean Elevation Data (DMED) from Digital Terrain Elevation Data (DTED) already available to DCOAD. An importer was written to import the DMED data and a new DMED shared service was created to maintain the data. The DMED ISR coverage estimator takes the results of the standard ISR estimator and modifies each block returned by it based on the DMED data applicable to the block. DMED data is considered applicable to a block if its data cell overlap's the estimate block.

The probability for the estimate block is modified based on a heuristic that uses the standard deviation in elevation from the DMED data. The heuristic computes a "terrain effect" constant for an estimation block based on the average standard deviation from all DMED blocks that apply to an estimation block (Note that an estimation block can be a different size from a DMED

block and hence could overlap multiple DMED blocks). The “terrain constant” is computed using a logistic curve function of the following form:

$$genLogistic(x) = A + \frac{C}{\left(1 + T \cdot e^{-B \cdot (x - M)}\right)^{1/T}}$$

where the constants, A, C, M, B, and T are defined as:

$$A = 0$$
$$C = 1$$
$$M = 200$$
$$B = \frac{1}{100}$$
$$T = \frac{1}{20}$$

The terrain effect is computed as:

$$terrainEffect(x) = 1 - genLogistic(x)$$

where the variable x is the average standard deviation. This function has the following plot:

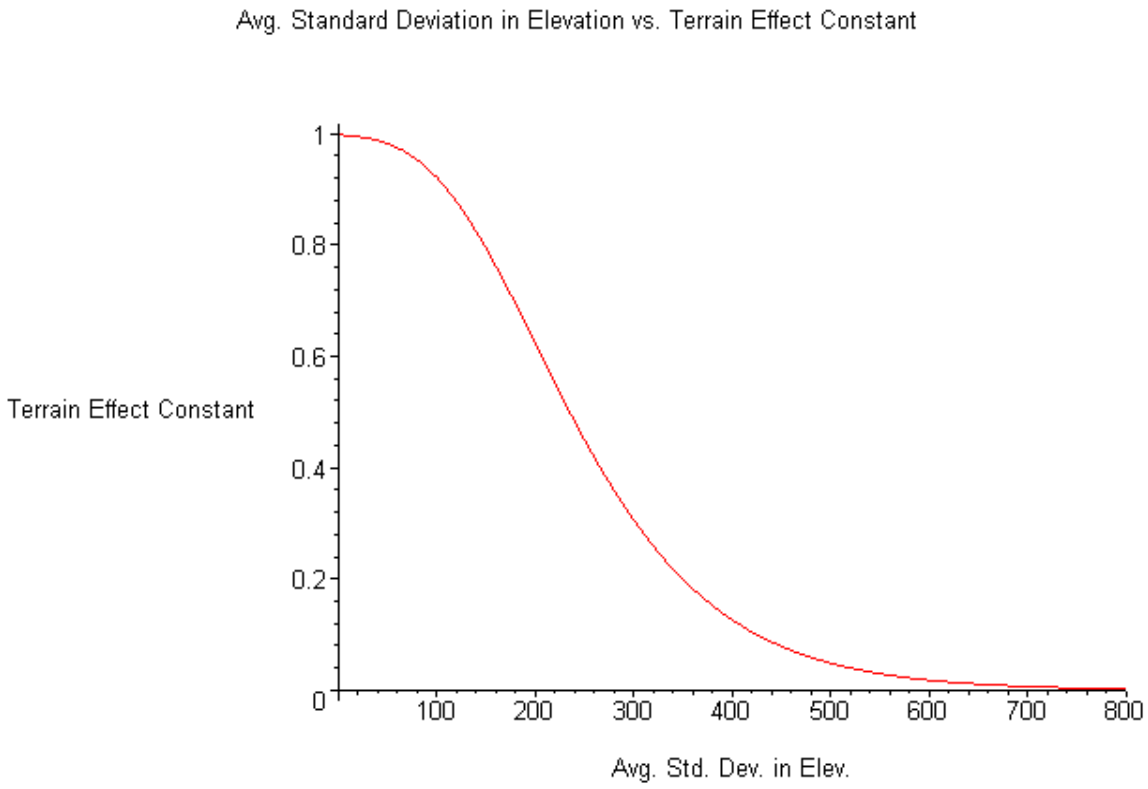

Figure 5 - The terrain effect constant as a function of the average standard deviation in elevation over all DMED data blocks that apply the estimation block.

The resultant probability for an estimation block is then computed as:

$$P'(x) = P(x) * terrainEffect(s)$$

P'(x) is the resultant probability for the estimation block with number, x, P(x) was the original probability for the estimation block with number x (as computed by the standard ISR coverage estimator), and s is the average standard deviation in elevation from all DMED blocks that apply to the estimation block.

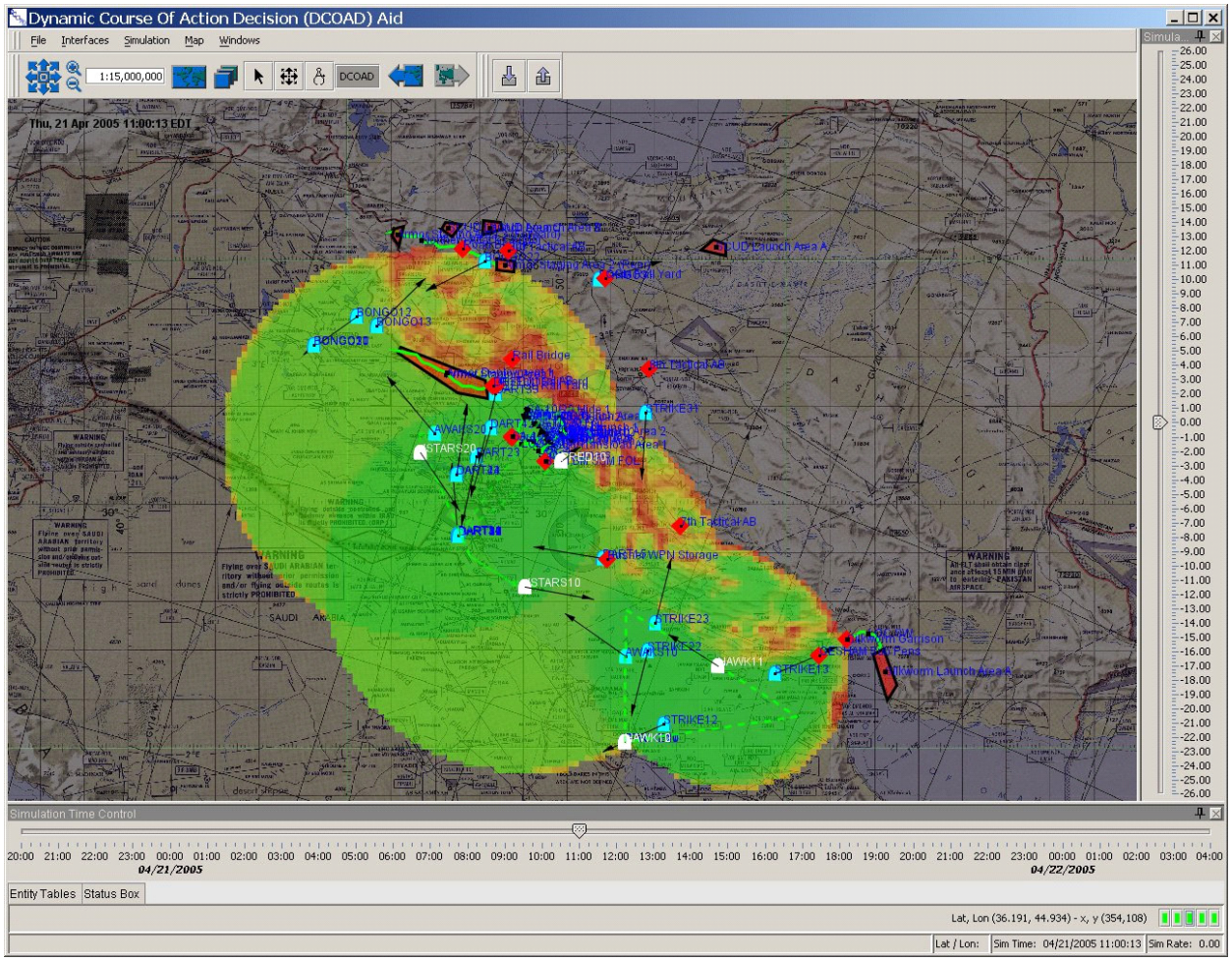

Figure 6 - Depiction of the ISR coverage estimate taking terrain masking into account for the entire Iranian scenario. Terrain masking is performed using the standard deviation from DMED data.

4.1.3. Strike asset coverage model

The strike effectiveness model is similar to the ISR collection model, except that sensor coverage and sensor performance are replaced by weapon range and munitions effectiveness, respectively. Once retasking opportunities are analyzed, a footprint for weapon range is determined based on the weapon type, and then munitions effectiveness is evaluated within the footprint. Munitions effectiveness depends on the type of munitions and type of target. This information is derived from a database called the Joint Munitions Effectiveness Manual (JMEM).

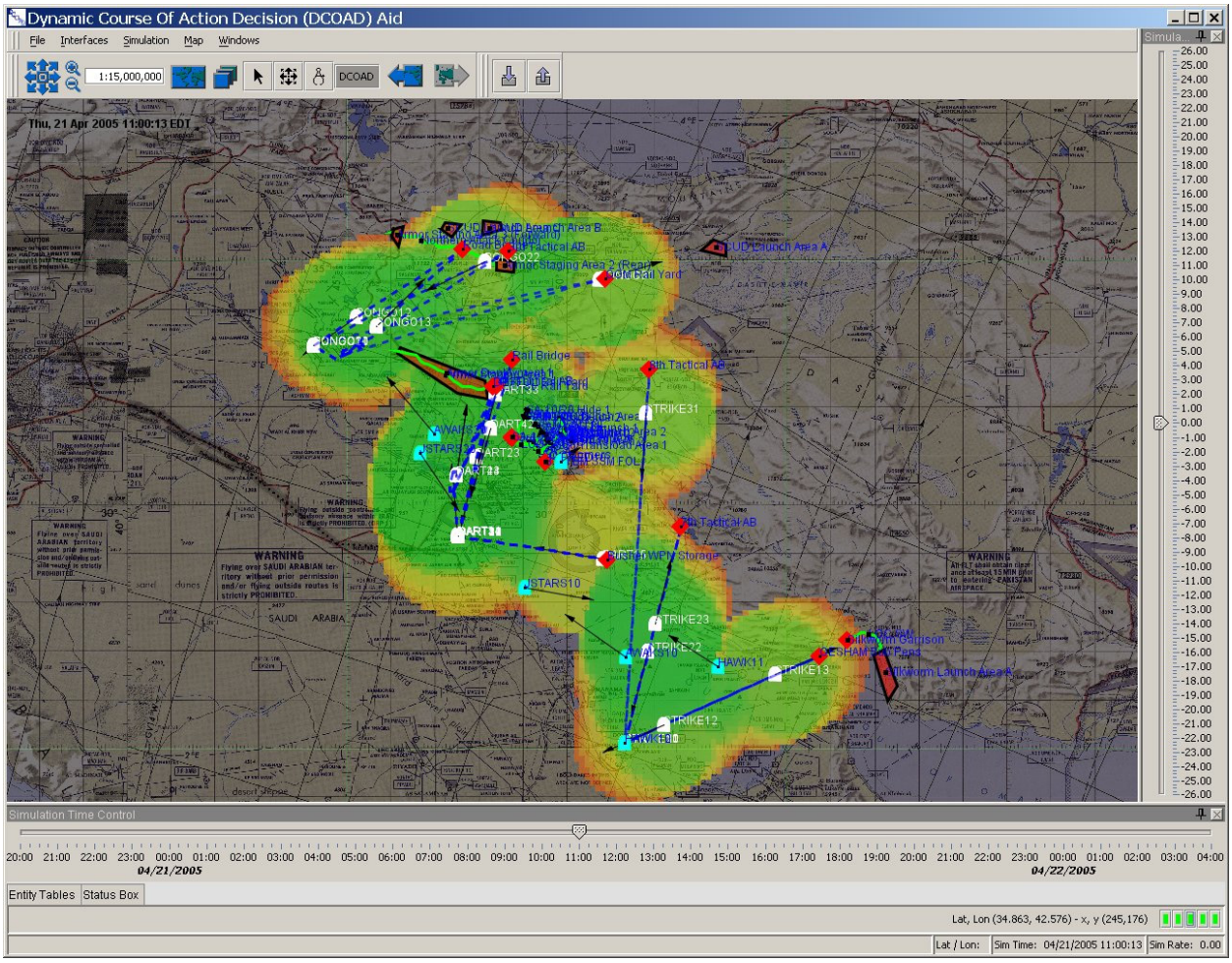

Figure 7 - Depiction of the strike coverage estimate for the entire Iranian scenario.

4.1.4. Composite model

We have discussed three different models that provide information in support of PBA for TSTs. Individually; these models evaluate the likelihood of TST occurrence, the coverage of our ISR assets, and the coverage of our strike assets. Combining these models provides additional PBA information. For example, combining ISR asset coverage with strike asset coverage provides operators with an assessment of where they are likely to be able to find and prosecute TSTs. If the TST occurrence model is included, operators have a single depiction of potential "hot spots" where strike and ISR coverage are insufficient for anticipated TST activity.

4.1.4.1. TST coverage model

The ISR asset coverage model and the strike asset coverage model are combined to create a TST coverage model that identifies regions in the battlespace for which there is sufficient strike and ISR coverage. This model is used to determine whether current strike and ISR assets are

sufficient to address TST "pop-ups" anywhere in the battlespace. To construct the TST coverage model, we defined two performance metrics to be used in the ISR asset coverage model and the strike asset coverage model. First, we define a sensor performance metric P_{FIX}, that corresponds to the probability that an ISR asset can detect and maintain track on a notional TST. Second, we define a strike effectiveness metric P_{STRIKE}, that corresponds to the probability that a strike asset can acquire (e.g., via a hand-off from the ISR assets) and prosecute a notional TST. When these metrics are used in the ISR asset coverage model and the strike asset coverage model, we can then combine these models as follows:

$$P_{COVERAGE} = P_{FIX} \cdot P_{STRIKE}$$

where $P_{COVERAGE}$ corresponds to the probability that a "pop-up" TST can be found, fixed, tracked, targeted, engaged and assessed. P_{FIX} and P_{STRIKE} are simply the probabilities of ISR and strike coverage as computed by the ISR and strike estimators described above. Limited resources typically imply that TST coverage can not be uniformly high across the battlespace. Next we show how all three models were combined to help operators tailor TST coverage based on predicted TST occurrence.

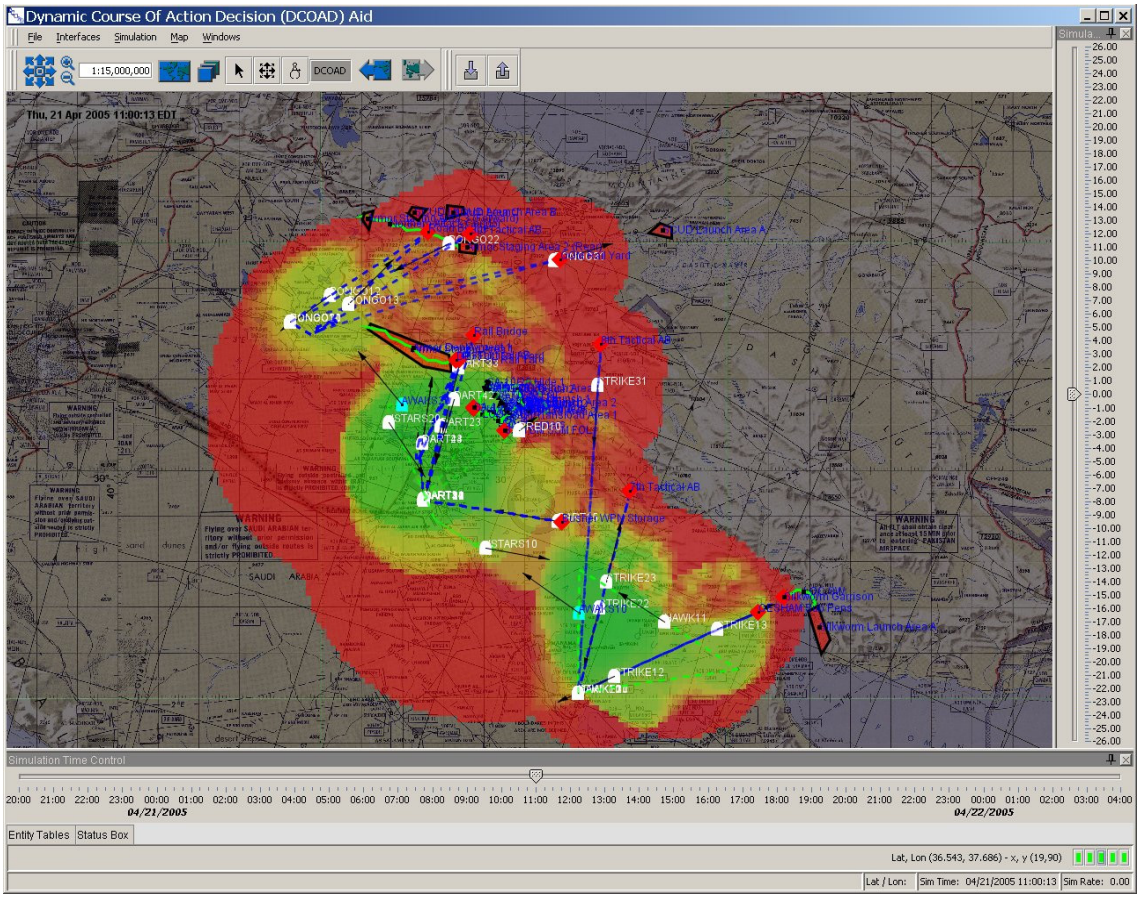

Figure 8 - Depiction of the composite ISR and strike coverage for the entire Iranian scenario. Note that the ISR DMED estimator was used in this composite estimator.

4.1.4.2. TST gap model

The TST gap model includes the TST occurrence model, the ISR asset coverage model and the Strike asset coverage model. By including TST occurrence, the TST gap model focuses on those regions in which TST activity is likely, while highlighting where TST coverage is insufficient (i.e., ISR and or strike coverage may be inadequate). The TST gap model uses the probability of TST occurrence PTST and the probability of TST coverage $P_{COVERAGE}$ to determine P_{GAP}:

$$P_{GAP} = P_{TST} \cdot \left(1 - P_{COVERAGE}\right)$$

where P_{GAP} is the probability that a TST will occur where there is insufficient coverage.

The TST coverage model and the TST gap model are two examples of how DCOAD combines its three basic models to provide operators with PBA for TSTs. Additional models could be developed to address specific PBA request, or consider additional factors. For example, a composite model could be generated to account for the specific temporal requirements associated with the time sensitive targeting process.

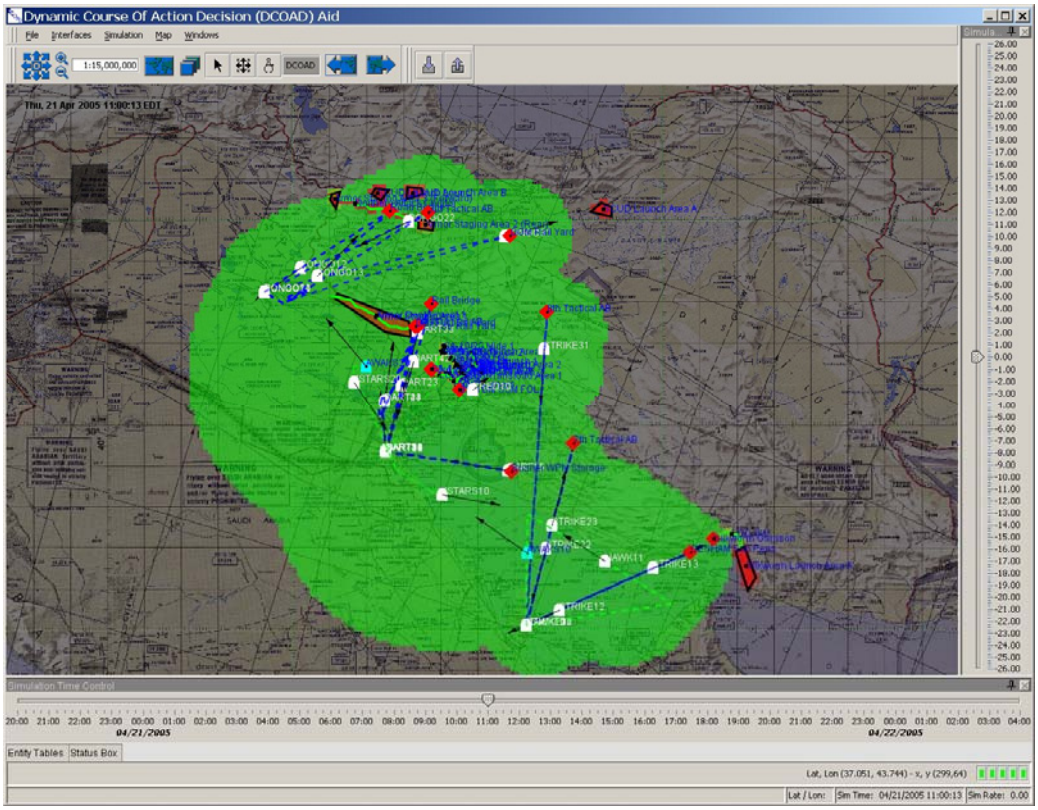

Figure 9 - Depiction of the composite gap estimate for the entire Iranian scenario. Note the gaps present at locations to the southeast and northwest. There are also gaps present on the east side of the central A2IPB target area. These correspond to areas of likely TST activity where the ISR or strike coverage is low.

4.1.5. DMED standard deviation coverage

The DMED standard deviation coverage is not an estimate, per se, in that it does not actually perform estimation. In fact, the values associated with each block from which the colors were computed are not probabilistic, but rather measures of standard deviation. The estimator was developed for two reasons; to demonstrate the flexibility of the DCOAD estimation framework and as a testing tool for the ISR coverage estimator with DMED.

The blocks displayed by this estimator are the same size as the blocks used in the DMED data. The colors for the blocks are computed from the standard deviation in elevation for the block. A red block corresponded to the highest standard deviation in the data set and a green block corresponded to a 0 standard deviation. Note that as with all DCOAD estimators, these colors and the ranges associated with them can be adjusted by the operator. Upon mousing over a block from this estimator, the detailed DMED information is displayed in a tool tip further demonstrating the flexibility of the estimator framework.

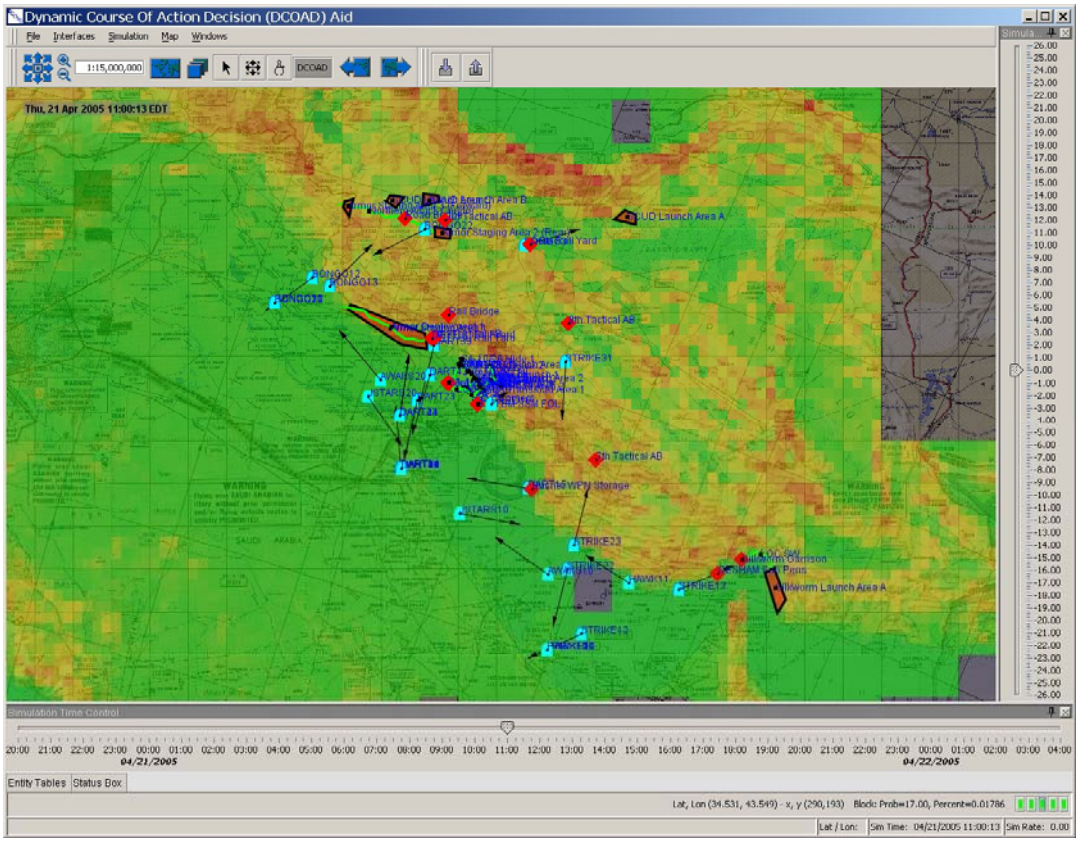

Figure 10 - Depiction of the Digital Mean Elevation Data over the campaign area.

4.1.6. Dynamic estimator updates from tracks

The DCOAD estimation framework does not impose limitations on the source of input from which an estimator can compute its estimate. We have taken advantage of this fact to provide

estimators that dynamically update probabilistic estimates based on near real time track input from the Tactical Management Service (TMS). TMS input is utilized in three estimators; the ISR coverage estimator, the strike coverage estimator, and the dynamic TST estimator. The ISR and strike coverage estimators essentially process tracks in the same manner as the static estimate, but the dynamic TST estimator algorithm is more complicated. Both are explained in more detail below.

4.1.6.1. Dynamic ISR / Strike estimation

The concept for the modifications to the ISR and strike estimation algorithms is relatively simple. The algorithm for calculating the ISR and strike coverage remains unchanged, but the path followed by the asset is modified based on track inputs. The algorithm simply uses this dynamic path instead of the planned path to generate its coverage estimates. Specifically, when an asset receives track information, its path is changed to a new path that uses both the planned path and the track history for the asset. This new path computes the dynamic parameters (position, heading, course, and speed) for an asset at time t as follows. If t lied within the time window for which the entity has tracks, the parameters are computed based on a linear interpolation between the two tracks nearest in time to t that encompass t. E.g.,

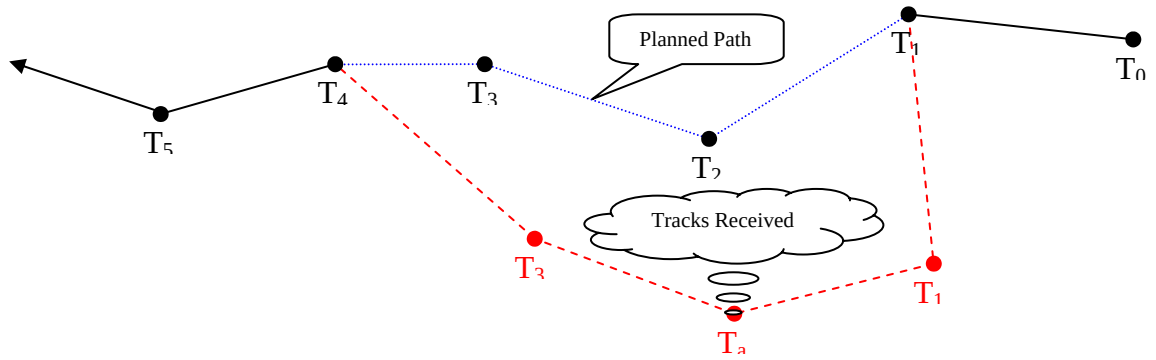

Figure 11 - Depiction of the effect of dynamic updates on ISR and strike coverage estimation.

This shows how the simulated path is affected by tracks associated with an asset. The red dotted line is the part of the path that would be followed for any time period, t, such that T1 < t < T4. Outside of that time period, the original planned path in black is followed. As more tracks are received, this picture changes. Since the estimate for both ISR and strike coverage is based on the path for the asset, the estimator uses the new path taking track input into account.

4.1.6.2. Dynamic TST estimation

The algorithm for dynamic TST estimation has three primary steps. They are as follows.

1. Given a track, the possible Lines of Communications (LOCs) and Infrastructure Areas that could be associated with the track are found along with the probability of the association.

2. Next, the possible TST units that could be associated with the track are found. These TST units must also be associated with a LOC or Infrastructure area in the A2IPB input.

3. Finally, the probabilities of association for each TST unit with an A2IPB product are updated based on the probable associations with track data computed in 1 and 2 above.

4.1.6.2.1. Identifying LOCs / Infrastructure Areas ↔ Track associations

We identify which LOCs or Infrastructure Areas to associate with a track as follows:

1. We find the closest distance from the track to every LOC and Infrastructure Area.
2. We compute the average and standard deviation of the distances.
3. Any LOC or Infrastructure Area that is within n standard deviations from the average is associated with the track. n was a configurable floating point parameter for the estimator. For each A2IPB product that meets the above condition, we set the probability of the association to be

$$\mathbf{P}(t_i, l_k) = \left(1.0 - \frac{d(t_i, l_k)}{\sum_{l_n \in Matched\ Prods} d(t_i, l_n)} \right)$$

Figure 12 - Probability of a track, t_i being associated with an A2IPB product, l_k. d (t, l) is a distance function (e.g., Euclidean distance).

i.e., it's just 1.0 minus the percentage of the distance between the track and an A2IPB product out of the sum of all of the distances from that track to all A2IPB products that meet the above condition.

The track / A2IPB product associations are more concisely represented in matrix form. This matrix is called the $\mathbf{P_{TL}}$ matrix (track ↔ LOC[1] association matrix).

[1] This is for all A2IPB products, but we just used L since using P for products would confuse things as P is usually used to represent a probability.

$$\mathbf{P_{TL}} \quad \begin{matrix} & L_1 & L_2 & \dots & L_n & N \\ T_1 & 0.2 & 0.3 & \dots & 0.1 & 0.0 \\ T_2 & 0.1 & 0.4 & \dots & 0.0 & 0.2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ T_m & 0.1 & 0.0 & \dots & 0.1 & 0.5 \end{matrix}$$

In this matrix form, the rows (T_1 … T_m) represent the m tracks, the columns (L_1 … L_n) represent the n A2IPB products, and the values in the matrix are the probability of association between the track and A2IPB product for the row and column on which the value lied. The last column, labeled N, represents the probability of the track not being associated with any A2IPB product.

4.1.6.2.2. Identifying Unit ↔ Track associations

We identified which units to associate with a track using mappings between category codes, platform type, and specific type. Track attributes include either a platform type and/or a specific type which we will generically call the target type for purposes of this description. The track specific type is used when provided. Otherwise, the algorithm defaults to using the platform type. We reverse map this target type to all category codes that can map to the target type. Any units that have one of these category codes is considered as possibly associated with the track. This is also represented in matrix form as follows. This matrix is called the $\mathbf{P_{TU}}$ matrix (track ↔ unit association matrix).

$$\mathbf{P_{TU}} \quad \begin{matrix} & U_1 & U_2 & \dots & U_n & N \\ T_1 & 0.3 & 0.3 & \dots & 0.0 & 0.3 \\ T_2 & 0.3 & 0.0 & \dots & 0.3 & 0.3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ T_m & 0.2 & 0.2 & \dots & 0.2 & 0.2 \end{matrix}$$

The $\mathbf{P_{TU}}$ matrix is basically the same as the $\mathbf{P_{TL}}$ matrix except that the columns represent units. Each entry in the matrix represents the probability of the track in the corresponding row (T_1 … T_m) being a track for the unit in the corresponding column (U_1 … U_n). The N in the last column represents the probability that a track does not associate with any unit i.e., the probability that the track was a new TST object. This was implemented to uniformly distribute the probability mass over all objects that could possibly associate with the track (based on category code) and over the track not associating with any unit. Note that all rows are normalized to sum to 1.0 as we assume that a track must correspond to some object (which may not be a known unit).

4.1.6.2.3. Computing the Unit ↔ A2IPB Product associations

The last step is to update the Unit ↔ A2IPB product associations based on our previous estimates of these associations and the new information received from track input. To do this

efficiently, we represent the Unit ↔ A2IPB product associations in matrix form as follows. This matrix is called the $\mathbf{P_{UL}}$ matrix (unit ↔ A2IPB product association matrix).

$$\begin{array}{c} \mathbf{P_{UL}} \\ U_1 \\ U_2 \\ \vdots \\ U_m \\ M \end{array} \begin{array}{c} \begin{array}{ccccc} L_1 & L_2 & \cdots & L_n & N \end{array} \\ \begin{pmatrix} 0.2 & 0.1 & \cdots & 0.3 & 0.0 \\ 0.3 & 0.4 & \cdots & 0.1 & 0.0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0.1 & 0.2 & \cdots & 0.3 & 0.0 \\ 0.5 & 0.2 & \cdots & 0.1 & 0.0 \end{pmatrix} \end{array}$$

The rows correspond to units and the columns correspond to A2IPB products. Each entry in the matrix represents the probability of the unit in the corresponding row (U_1 … U_m) following (or heading towards) the A2IPB product in the corresponding column (L_1 … L_n). The N in the last column represents the probability of a unit being off road. The M in the last row represents the probability of a new TST occurring. Initially, the last row was uniform across the A2IPB products (since, if a new TST occurred, we wouldn't know where). But, as we get track information, this corresponds with where tracks are coming in. Note, that as with the $\mathbf{P_{TL}}$ and $\mathbf{P_{TU}}$ matrices, the rows are normalized to sum to 1.0 as a unit must either be associated with an A2IPB product or must be off road.

With this model, we compute a new $\mathbf{P_{UL}}^{\mathbf{new}}$ using the track information received during the time slice for which the computation was performed. This was:

$$\mathbf{P}_{\mathbf{UL}}^{new} = \left(\mathbf{P_{TU}}\right)^{\mathbf{T}} \cdot \mathbf{P_{TL}}$$

$$\mathbf{P}_{\mathbf{UL}}^{new\_norm} = normalize\left(\mathbf{P}_{\mathbf{UL}}^{new}\right)$$

where $\mathbf{P_{TL}}$ and $\mathbf{P_{TU}}$ are as defined in the above paragraphs, $\mathbf{P_{UL}}^{\mathbf{new}}$ and $\mathbf{P_{UL}}^{\mathbf{new\_norm}}$ are the new $\mathbf{P_{UL}}$ and normalized $\mathbf{P_{UL}}$ matrices respectively, and $(\mathbf{P_{TU}})^{T}$ is the standard transpose operator applied to the $\mathbf{P_{TU}}$ matrix. The normalize function scales a row such that it sums to 1.0 for each row in the matrix. Normalize is actually implemented as a matrix operation by computing the sums of the rows, S, placing the inverse of the sums on the diagonal of a new matrix, N, and computing as shown below:

$$\mathbf{S} = \mathbf{P}_{\mathbf{UL}}^{new} \cdot \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_m \end{bmatrix}$$

$$\mathbf{N} = \begin{bmatrix} \frac{1}{S_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{S_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{S_m} \end{bmatrix}$$

$$\mathbf{P}_{\mathbf{UL}}^{new\_norm} = \mathbf{N} \cdot \mathbf{P}_{\mathbf{UL}}^{new}$$

We incrementally update $\mathbf{P_{UL}}^{\mathbf{new\_norm}}$ using the non -normalized result and incorporating new track data. This is done as:

$$\mathbf{P}_{\mathbf{UL}}^{new\_norm} = \mathbf{N} \cdot \left(\mathbf{P}_{\mathbf{UL}}^{new} + \left(\mathbf{P}_{\mathbf{TU}}^{new} \right)^{\mathbf{T}} \cdot \mathbf{P}_{\mathbf{TL}}^{new} \right)$$

where $\mathbf{P_{TU}}^{\mathbf{new}}$ and $\mathbf{P_{TL}}^{\mathbf{new}}$ are the same form as described above but only contain the new track entries. These are normalized as described above prior to the computation.

4.1.7. Verification and Validation

Although the algorithms used in DCOAD for the computation of probabilistic values were not validated through a formal verification and validation process it would be possible to perform such validation on the mathematical models used by the estimators.

For instance, the strike asset coverage and ISR asset coverage models are both based on the physical characteristics of the aircraft and the ability for the aircraft to perform its functions (e.g., destroy or detect) against a given target type. The probabilities for destruction and detection are not directly computed by the algorithm, but are instead retrieved from “parametric data” which is populated with the appropriate probabilities (e.g., from JMEM for strike assets). Hence, the model itself is a valid model since it is based on validated data sources. However, the algorithms’ do handle combining these probabilities for multiple assets over an area and over time. The mathematics behind these combining computations (which are described in the sections for each estimator above) could be validated through inspection by one versed in probability theory.

The same validation argument may be made for the target occurrence estimator. This estimator uses IPB data for the computation of the target probabilities and the probabilities are explicitly based on the likelihood of occurrence of targets from IPB. Hence, to the extent that the IPB data is valid, the model represented by the estimator should be valid. The only verification necessary would be on the mathematics combining probabilities over space and time for the IPB data.

Likewise, the model behind the composite estimators simply uses joint probabilities over the results from the sub-estimators. Hence, the model should be valid under the assumption that the sub-estimators are valid. Verification would simply consist of inspecting the mathematical equations and the resultant implementation in software for errors.

4.2. System Architecture

The DCOAD architecture consists of three major components; the graphical user interface (GUI), the shared data services, and the estimator framework. These fit into the paradigm of the model-view-controller (MVC) architecture used in most graphical applications. The map and tables (view) depict the data from the estimators in the estimation framework (model) using the shared data services (controller). We will cover each of these components in more detail below.

4.2.1. Graphical User Interface (GUI)

DCOAD is a map and table based application. DCOAD provides standard map controls and overlays via the OpenMap® map toolkit (see GUI snapshot at Figure 13). This includes zooming, a browser-like projection stack, map overview, imagery layers, and other features. In addition to the standard map controls and overlays, DCOAD provides a means of depicting probabilistic information discretized over space and time as an overlay on the map. This is our primary means of providing PBA to the operator. The tabular information supplements the map with details for each of the objects depicted. DCOAD has the ability to run either in a real-time execution mode (in which real-time track data provides updated positional information) or in a simulation mode in which a planned scenario is flown out. In the planning mode, the simulation time can be controlled via a simple time and rate slider. Dockable internal windows are used in DCOAD to maximize usability and customization of the interface.

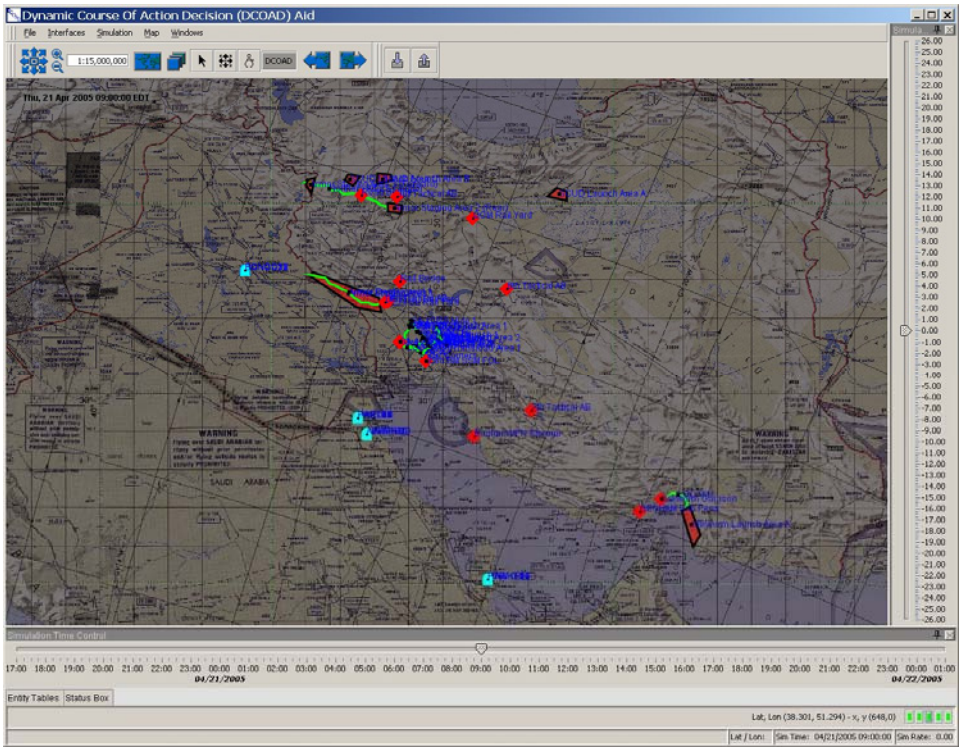

Figure 13 - Snapshot of the primary DCOAD user interface.

4.2.2. Shared data managers

DCOAD implements a service-centric architecture. Access to both data and general interfaces is provided via singleton services. Data is imported from external databases (or interfaces) and transformed into a data format that is usable within the system. The transformation is modularized such that the input to the transform is the external data source and the output of the transform is a singleton shared data service. All shared data services implement a common interface. The interface provides methods for registration of event handlers (e.g., for notification when objects are added, modified, or removed from a service) and methods for adding, updating, and removing objects from a service. An abstract service class implements this interface and provides the standard implementation for parts of the interface available in all services.

As mentioned above, the shared services have an event notification framework defined in the interface. A subset of events are defined for the base shared service hierarchy (e.g., an event for a data object modification) while other events are created for specific services. Callbacks are registered to run either synchronously or asynchronously with respect to the thread of execution triggering the callback. Synchronous callbacks are executed in the same thread of execution while asynchronous callbacks are queued and executed in the event handling thread [Not the same as Abstract Windowing Toolkit (AWT) events to keep from hanging the GUI]. The

intention is that events requiring heavy computation (i.e., estimate calculations) execute asynchronously while light computation events execute synchronously.

The data managers are defined as singleton services accessible to the entire DCOAD system. This naturally leads to the question of how to maintain data integrity in a multi-threaded application. DCOAD uses a coarse grained read/write locking policy where multiple services can be locked simultaneously for the extent of a read/write transaction. The extent of the transaction is determined by the module accessing the service. Import transformations write lock a service, or multiple services, for the duration of the data import while estimators and other internal modules read lock any services being accessed for the duration of their access. The lock interface guarantees that the order of locking for multiple services is constant over multiple calls to prevent deadlock conditions.

4.2.2.1. External interfaces

DCOAD requires access to several external interfaces to provide information on which to base TST occurrence estimates and ISR and strike capabilities. All DCOAD external data interfaces are designed as transformations from the external data representation to internal Java object data representations. In cases where the external data representations are Java objects and these objects are accessible, the transformation is simply the identity unless special circumstances required a different format. Data objects requiring local storage via a cache (e.g., friendly order of battle) are managed by a shared service. Data objects that are not cached locally (e.g., tracks), are simply transformed and passed to processes or methods that handled the data (e.g., updates to data objects in another shared service). These naturally fell into categorization as either stateless or state full data interfaces. Below is a brief description of the external interfaces used in DCOAD.

Air Operations Database (AODB): DCOAD imports data from the Theater Battle Management Core System (TBMCS) AODB for purposes of displaying and utilizing the planned strike asset allocation for the campaign. This provides the strike asset coverage and weapons load information used in the Strike Asset Coverage Estimator, TST Coverage Estimator, and TST Gap Estimator. The Air Battle Plan (ABP) is flown out using the simulation capabilities of DCOAD to provide further situational awareness to the operator.

Automated Assistance with Intelligence Preparation of the Battelspace (A2IPB): DCOAD imports products from the A2IPB tool via XML file import. Virtually all data defined in the A2IPB schema is imported and can be displayed both on the map and in tabular form to provide further situational awareness to the operator. Only Infrastructure Areas and Lines of Communication are currently used for estimation purposes.

Intelligence, Surveillance, and Reconnaissance (ISR) Input: DCOAD supports an interface to a XML based flat file format for retrieval of the ISR plan. The flat file includes both ISR and strike assets and is known as a “Mission File”.

Tactical Management Services (TMS): DCOAD receives real-time track updates for both friendly and hostile tracks via TBMCS Tactical Management Services (TMS). The TMS interface is a stateless service that propagates updated information from incoming tracks to the appropriate shared service maintaining the object to which the track is associated. Updates are processed in batch to prevent thrashing of the system. Thresholds are also utilized to determine when an update needs be applied or ignored.

Parametric Data (PD): Parametric data contains static information needed for DCOAD to perform estimation. This includes such items as aircraft performance information, target type associations, airbase data, weapon data, sensor data, weapons effectiveness data, and sensor effectiveness data. DCOAD supports an interface to a XML-based flat file format for retrieval of this information.

Digital Terrain Elevation Data (DTED): DCOAD imports DTED data via a standard import interface provided by the OpenMap map framework. This data is only used in the DTED overlay which is also an OpenMap product. The interface imports the DTED data from directories specified in the DCOAD properties file.

Digital Mean Elevation Data (DMED): DCOAD imports DMED data from DMED text files specified in the DMED configuration menu item. The DMED data is stored in a text file using a very simple comma-delimited format. These text files are included with standard DTED data distributions.

Compressed ARC Digital Raster Graphics (CADRG): DCOAD imports CADRG data via a standard import interface provided by the OpenMap map framework. This data is only used in the CADRG overlay which is also an OpenMap product. The interface imports the CADRG data from directories specified in the DCOAD properties file.

User Preferences: DCOAD uses a Java framework for persistent storage of user preferences data. During the development of DCOAD, a custom engine was written for this framework to store the persistent data in a directory structure rather than using the standard registry based implementation provided by Java. This avoided problems involving user permissions with regard to the operating system’s registry (This was done in response to an error that occurred while testing DCOAD in an experimental AOC). The user preferences may be exported from DCOAD into an XML flat file format and also imported from this same format. This allows the user preferences to be migrated from machine to machine or user to user.

4.2.3. Estimator framework

The estimator framework is a pluggable framework for calculating discretized estimates and representing the estimates in a manner that can be displayed either pictorially or output to external sources. The estimators are based on the concept of block shapes which will be described in more detail below. Estimators are dynamically loaded into the system via the estimator manager which is a singleton service implementing the shared services common interface. The estimator manager additionally allows an active estimator to be selected from the list of all loaded estimators. This is the estimator linked to the probability overlay for display on the map.

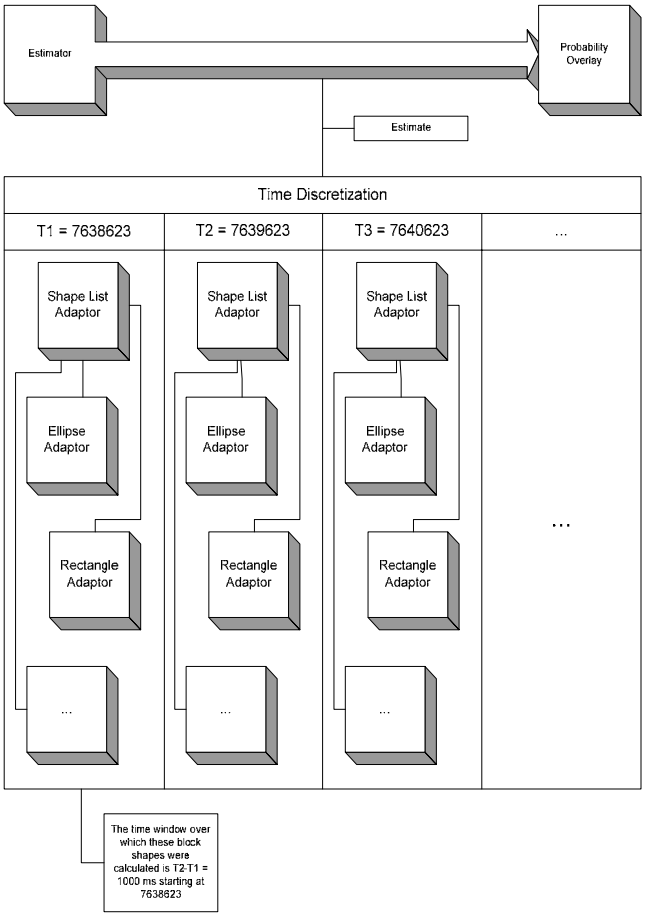

Figure 14 – Depiction of an estimate passed from an estimator to the probability overlay. The estimate may also be sent to an output transformation that will convert the estimate into an XML file for use by other applications.

Every estimator must implement the Estimator interface which defines the basic functionality for all estimators. This interface requires methods for enabling/disabling the estimator, retrieval of the current estimate, forced recalculation of the estimate, and similar functions. Additionally, the interface requires a method for retrieving a configuration panel widget so that model-specific parameters for the estimation algorithm can be set.

Each estimator is responsible for importing any data required for calculation of the estimate which could involve using the output from another estimator. The estimators implemented for DCOAD use the shared data managers to retrieve data used in computation of the estimate. The computed estimate is returned either via a function call or via a callback registered on the estimator. The format of the estimate, depicted in Figure 14, is a map keyed on a timestamp for the beginning of a time “bucket” where the value is a block shape as described below.

4.2.3.1. Block shapes

Block shapes are the means by which a probability distribution over space is discretized and represented. A block shape is a wrapper around a java shape with an associated “filler function”. The java shape is required to be one in which the coordinates for the shape are represented as doubles so that the coordinates can represent latitudinal (y axis) and longitudinal (x axis) radian values. The filler function is a function defining a mapping between discretized blocks within the java shape and a probabilistic value. Each shape implementing the block shape interface provides a function for “rendering” the block shape. This render function takes a discretizer as a parameter which provides a model for a discretization of the earth based on a given granularity. The “render” function uses a shape-specific algorithm to determine the blocks from the discretization model that cover the real-valued java shape, and then assign a probability for each of the discretized blocks using the filler function. Multiple block shapes can be aggregated via a block shape aggregate. A blending function is provided with an aggregate shape defining how the probabilities from each block shape are combined. With this framework, an author of an estimator need only to generate real-valued shapes and corresponding filler and blending functions to generate an estimate over space that works at any granularity.

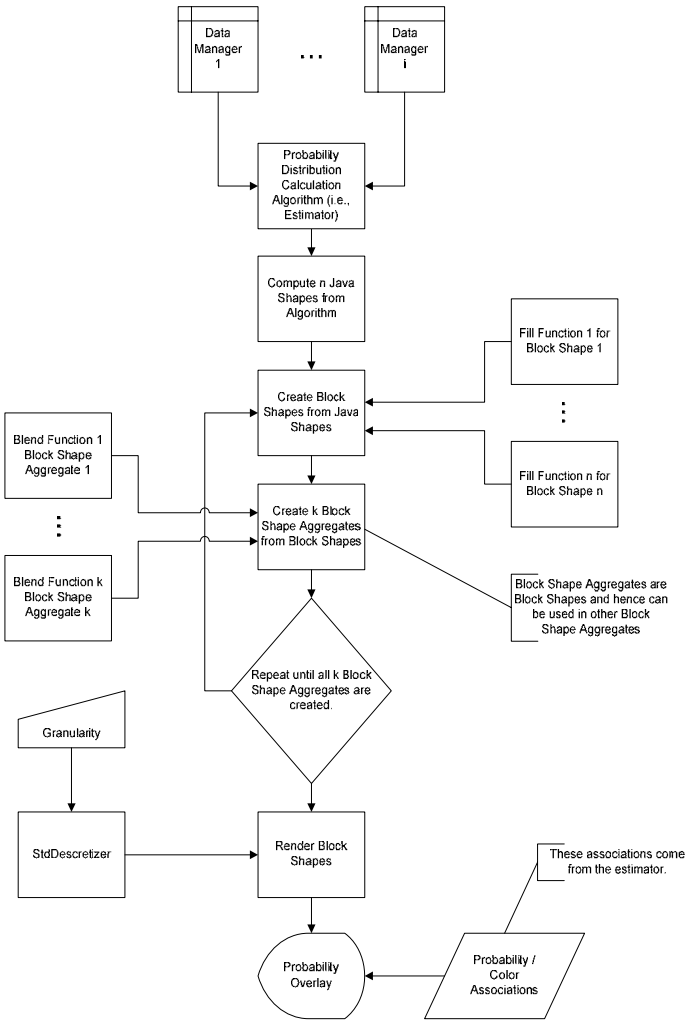

Figure 15 - Depiction of the "pipeline" for rendering the probability distribution from an estimator.

Figure 15 depicts the entire "pipeline" used by the estimator framework starting at the shared data managers and completing at the probability overlay. The active estimator imports data from shared data services and processes it to generate real-valued Java shapes that represent the area of the appropriate probability distribution. A fill function is assigned to each shape representing the values in the distribution. The algorithm can generate multiple shapes; hence the use of multiple fill functions. These shapes are aggregated, possibly into multiple aggregates depending on the area of estimation, using blending functions to merge the discretized probabilities from the aggregated shapes. Finally, when all block shapes are defined, the shapes are rendered (i.e., discretized) with a given granularity and sent to the probability overlay. The probability overlay displays the discretized blocks with the appropriate color using the probability / color associations from the estimator.

4.2.3.2. Discretization in space

The lowest level of the estimator design is the discretizer. The purpose of the discretizer is to partition the earth into equal sized enumerated chunks such that for given a latitude/longitude coordinate, the discretizer returns a unique block number. Furthermore, given a block number, the discretizer returns the coordinates for the corners of the block and the center point for the block. One of the more powerful features of the discretizer is the ability to return a "view" of a subset of the discretized space. A view is returned over a given bounding box within the space. The blocks within the view are accessible as if they were in a two dimensional array with the lower left hand block at (0, 0) with increasing blocks values to the north and east. A translation function is provided that maps the local block tuple (as a row/col offset) into a global block number. This made authoring "render" functions for block shapes easier.

4.2.3.3. Probability overlay

The probability overlay is the primary data display for DCOAD. Its purpose is to display the probabilistic information associated with an estimator. The overlay is automatically associated with the active estimator from the estimator service and receives changes of the active estimator from the estimator service. The probability overlay pulls estimate data from the active estimator upon notification of a change in the estimate. It is designed such that multiple estimates updated in succession result in only a single update of the overlay. The colors for the blocks in the overlay are calculated using probability / color associations provided by the estimator (This allowed different color schemes based on the active estimator). These associations are pairs consisting of a probability and an associated Alpha/Red/Green/Blue (ARGB) color value. The color of a block displayed on the overlay is calculated as an ARGB value that lies on a linear interpolation of the two closest probability / color pairs, as shown in Figure 16.

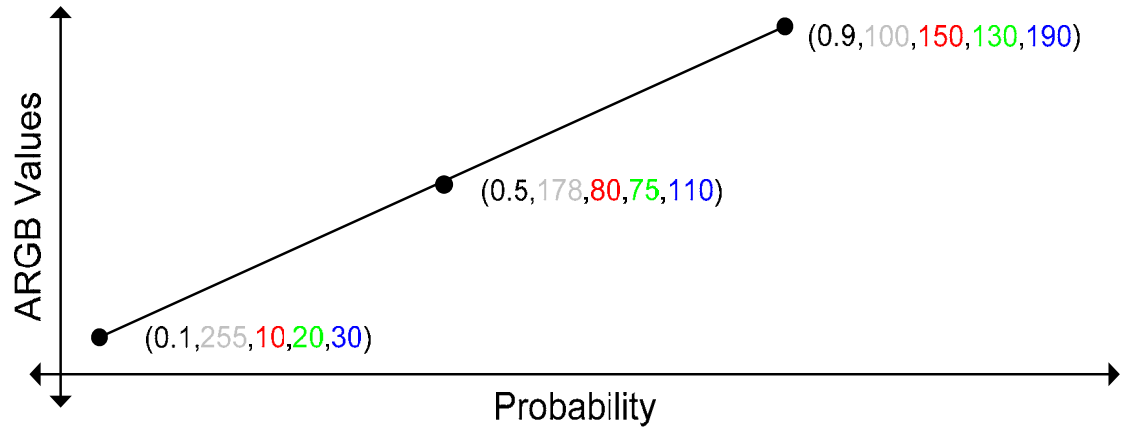

Figure 16 – Linear interpolation of ARGB value based on the input probability and two probability / color associations as input by the operator.

4.3. DCOAD Software Architecture

Each DCOAD software unit provides an additional piece of functionality to the system. There are three types of units used in the design of the system: Standalone Units, Architecture Units, and Modification Units.

Standalone Units are commonly known as modules. They provide a standalone piece of functionality with well defined inputs and outputs. These are the “black boxes” of the system.

Architecture Units provide a framework in which the Standalone Units work. In the case of DCOAD, the GUI Backbone, Map Entity Shared Data Services, Parametric Data Manager, and the Probability Estimators Base Functionality units provide the framework in which our modules work. These are essentially the “glue” modules.

Modification Units consist of modifications to a previously created unit to add new functionality. Note that a standalone unit can use another standalone unit so long as it does not change the functionality of the other unit. Otherwise, it would be considered a Modification unit.

DCOAD is divided into 18 individual units, categorized by the unit definitions provided above. They are listed below along with a brief definition of the functionality provided by each unit.

4.3.1. GUI Backbone (Architecture)

The GUI backbone unit provides the framework in which the operator interacts with DCOAD. It provides the main window which includes the map, map controls, map status toolbar, a control to adjust the time window for calculations, a text status window, and table to display detailed object information.

4.3.2. Map Entities Shared Data Services (Architecture)

The Map Entities Shared Data Services unit provides a service based framework for storing information. A base service class provides the common service functionalities such as addition, modification, and removal of shared objects and entry or removal of callback handlers on specific events. The services permit events to be fired in the implementation of other services derived from the base so that these callbacks can be called at the appropriate times. Additionally, both synchronous and asynchronous event models are supported.

4.3.3. Parametric Data Manager (Architecture)

The Parametric Data Manager unit provides the framework for (generally) static data that is used throughout the DCOAD system. This includes such data as the aircraft performance information, weapon speeds and ranges, sensor coverage areas and associated information, standard

conventional loads (SCLs), and other infrequently changing data. The parametric data services unit implementation includes an XML schema defining how all of the data is stored. From this schema, a set of representative data classes are auto-generated. The auto-generated data classes have the ability to import data objects from an XML file that is valid with respect to the XML schema. This allows such data to be easily represented and portable.

4.3.4. A2IPB Interface (Standalone)

The A2IPB Interface unit provides the primary probable TST occurrence information for use in the planning and execution phases of DCOAD. Manual loading of A2IPB products is initiated through the import menu option or toolbar button. The A2IPB interface imports the following products: COAs, Infrastructure Areas, Individuals, Units, LOCs, TAIs, and NAIs.

4.3.5. AODB Interface (Standalone)

The AODB Interface unit loads the planned strike missions for the day. To extend coverage over multiple days, multiple air battle plans can be loaded. This provides the primary source of strike asset coverage during the planning phase of DCOAD. The AODB interface import is initiated via the import menu option or toolbar button.

4.3.6. ISRM Interface (Standalone)

The ISRM Interface unit loads planned ISR missions for the day. This provides the primary source of ISR asset coverage during the planning phase of DCOAD. The ISRM interface import is initiated via the import menu option or toolbar button. ISR assets are imported via the DCOAD mission file.

4.3.7. Probability Estimators Base Functionality (Architecture)

The Probability Estimators Base Functionality unit provides the framework in which the pluggable probability estimators are built. This includes the space and time discretization model, the interfaces to estimators and estimates, the probability overlay, the block shape implementations, and the shape adaptor implementations. In addition, the estimator GUI abstract classes and interfaces are implemented in this unit.

4.3.8. TST Occurrence Estimator (Standalone)

The TST Occurrence Estimator unit provides the algorithm for calculating probability of TST occurrence over space and time. Details of the algorithm design are specified above.

4.3.9. ISR Asset Coverage Estimator (Standalone)

The ISR Asset Coverage Estimator unit provides the algorithm for calculating probability of ISR asset coverage over space and time. Details of the algorithm design are specified above.

4.3.10. Strike Asset Coverage Estimator (Standalone)

The Strike Asset Coverage Estimator unit provides the algorithm for calculating probability of strike asset coverage over space and time. Details of the algorithm design are specified above.

4.3.11. Composite TST Prosecution Estimator (Standalone)

There are two composite estimators associated with the Composite TST Prosecution Estimator unit. The Gap Estimator provides the algorithm for calculating the probability of finding and prosecuting a TST given the probabilities of predicted TST occurrence, ISR asset coverage, and strike asset coverage. i.e., it essentially computes the joint probability of ISR asset and strike asset coverage given a probability of TST occurrence over space and time. Details of the algorithm design are specified above.

The Composite Coverage Estimator provides the algorithm for calculating the probability of finding and prosecuting a TST regardless of predicted TST occurrence. i.e., it essentially computes the joint probability of ISR and Strike asset coverage. Details of the algorithm design are specified above.

4.3.12. TMS Interface (Standalone)

The TMS Interface unit provides access to tracks in the tactical management service from the TBMCS. Tracks are filtered and correlated with map entities that already exist in the system. If the track correlates, the correlated entity is updated with the information contained within the track.

4.3.13. Dynamic Estimators (Standalone)

The Dynamic Estimators unit extends the current target estimator to take real-time track data into account. Details of the algorithm design are specified above.

4.3.14. GUI Enhancements (Modification)

The GUI Enhancements unit modifies the GUI framework to allow all windows in the GUI to be dockable. It also allows the GUI state (i.e., the Main Form) to be saved. The main menu includes more control options and changed so that all menu items are operational. Finally, the GUI Enhancements unit updated OpenMap to version 4.6.2.

4.3.15. ISR Estimator Enhancements (Standalone)

The ISR Estimator Enhancements unit extends the ISR Coverage Estimator by taking Digital Mean Elevation Data (DMED) into account when considering ISR coverage. Rather than performing expensive line of sight (LOS) calculations, this estimator uses a novel approach involving the standard deviation in elevation. Details of the algorithm design are specified above.

4.3.16. Block Shape Enhancements (Modification)

The Block Shape Enhancements unit attempted to modify the block shape framework to allow "holes" to be specified in shapes. Specifically, a shape could be given shape masks that mask out probability blocks from the rendered shape. This would allow arbitrary shapes to be used in the estimators.

This unit was only partially completed. The partially completed implementation of this unit is stored in the D30-Block_Shape_Enhancements branch in the CM repository. It compiles, but is untested and has performance issues (both speed and memory). To complete the implementation, the Space Distribution should probably be changed back to the old method of storing probability blocks (i.e., using a map keyed on the block number) rather than the complicated bitmap implementation.

4.3.17. Strike Estimator Enhancements (Modification)

The Strike Estimator Enhancements unit proposed to modify the Strike Coverage Estimator to take the minimum weapon range into account when computing the estimate. Specifically, a torus would have been used for the estimate computation rather than a circle. Additionally, the minimum and maximum range of each weapon on a platform would have been used in the estimate rather than the min and max range over all weapons on the platform.

This unit was not completed. The completion of this unit depended upon completion of the Block Shape Enhancements above.

4.3.18. SDT Interface Mockup (Standalone)

The SDT Interface Mockup unit proposed creating a mockup of the SDT tree interface to demonstrate how SDT would depict information from DCOAD to indicate possible adverse effects to the Effects Based Plan (EPB).

This unit was not completed. Program resources were not sufficient to implement this demonstration capability. The program manager decided the value added for implementing the terrain model for ISR asset Pd calculations was higher than pursuing the demonstration GUI.

5. Key Successes

5.1. Development of an Extensible Estimator Framework

A common estimator framework was developed to simplify the use of multiple estimators with DCOAD. It was recognized early in development that this extensible framework could be used to allow estimators to be developed as modules and "plugged in" to the framework. This can be utilized in the future to rapidly demonstrate or prototype potential PBA estimation algorithms.

Furthermore, in an operational environment, it provides a means for rapid updating of estimation algorithms to address a new need of the operators.

5.2. Development of a Practical PBA Visualization Environment

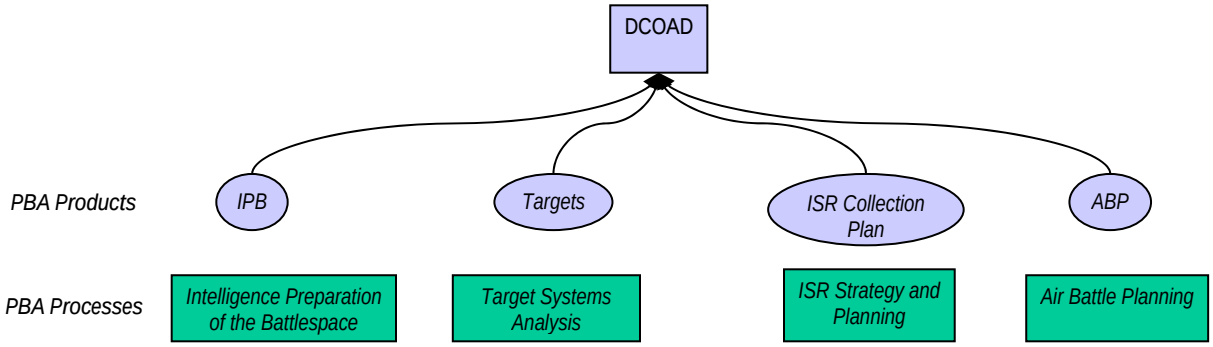

Figure 17 - DCOAD brings together all the products of the PBA process

As shown in figure 17, DCOAD is the first application that brings all the products of the PBA process (plus the daily air battle plan) to help operators predict and visualize:

- the likelihood of TST occurrence
- the likelihood of TST discovery
- the likelihood that, once discovered, TSTs can be successfully attacked, and
- the overall probability that TSTs can be successfully countered given the configuration of the daily air battle plan.

DCOAD's unique probabilistic overlays provide operators a practical, easy-to-interpret display that can be used to assess if the daily air battle plan is adequate to counter anticipated TST that might appear in the battlespace.

5.3. Development of a Novel Approach to Terrain Masking for ISR Estimation

To provide a more accurate ISR coverage estimate, DCOAD needed to take terrain masking into account during the estimation process. The traditional means of computing terrain masking for an ISR footprint is to perform Line of Sight (LOS) computations between the sensor and areas on the ground at specific angles relative to the ground and aircraft. This is a computationally expensive operation and is infeasible for near real-time probability estimates. If used, the complexity of such a computation would be further compounded by the fact that DCOAD must perform these computations for several predicted positions of the aircraft for a single estimate! This is due to the fact that estimates are computed over a period of time rather than at an instant in time.

Rather than using such complex LOS computations, a novel approach is used to address terrain masking in DCOAD. The approach involves the use of terrain variability in areas where coverage is to be computed. The concept behind the approach is that it is more difficult for a sensor to acquire and maintain track on a target in terrain that is more variable (i.e., more peaks and valleys) than in terrain that is less variable. From this concept, a heuristic was derived to vary the probability of ISR coverage based on the variability in terrain. The advantage in using this heuristic is that only local information is used (i.e., the standard deviation for the block for which the computation is being performed) rather than requiring data from surrounding blocks as well. This allows the computation to be computed in near real-time. This heuristic was not developed with any empirical data, and has not been validated in operational conditions. However, given empirical data, it is possible to tune the equations to yield a result similar to the use of traditional LOS computations. This could be accomplished through off line simulation. LOS measurements could be taken from selected ISR platform/sensors to targets in differing terrain conditions. ISR platforms would be deployed at normal operating altitudes using standard employment geometry. LOS measurements would be taken 360 degrees around the target area of interest. The probability of detection could be calculated for each terrain condition and compared to flat earth values. A terrain effect curve could then be plotted for each ISR asset.

6. Problems Encountered and Lessons Learned

6.1. Underestimated Development of GUI and Architecture Components

The development time and resources required to implement the GUI and architecture components of DCOAD were underestimated in the early phases of the project. This was caused by multiple factors listed below:

- Lack of a clear vision for the system in the initial design phase.
- Lack of familiarity with the mapping framework being used (OpenMap).
- Lack of familiarity using Java Swing (by the SW estimator).
- Uncertainty regarding the nature of some external interfaces and data necessary to support the design. For example, an actual ISR manager did not exist requiring a surrogate interface to be developed.

6.2. Spiral 3 Adjustments and Re-prioritization

Approaching the last year of the DCOAD program, it became clear that we were not going to be able to meet two of the four objectives for the program. These objectives included:

- A prioritization capability which dynamically accessed current and planned mission and target priorities based upon the impact those targets and missions have in influencing the effects-based plan objectives.
- A campaign assessment capability that assesses the impact that diverting missions to prosecute TSTs has on the campaign objective.

Our ability to dynamically prioritize current and planned mission and target priorities relied upon the capabilities provided by an application external to DCOAD. This application is the Causal Analysis Tool (CAT). CAT was developed to perform campaign level assessment and sensitivity analysis at the task level. DCOAD works at the target level [Desired Mean Point of Impact (DMPI)] and has a daily air battle plan focus. Although CAT demonstrated some experimental capability to conduct target level assessment, in general, it could not support the target level analysis needed to assess the impact of changing target and mission priorities on the effects-based plan. Further, individual target DMPIs have to be linked to objectives in the effects-based plan for an impact assessment to be conducted. No such linkage currently exists. As a result, we could not assess the impact of TST-generated changing target and mission priorities on the effects-based plan.

For DCOAD to assess the impact that diverting missions to prosecute TSTs has on campaign objectives, target-based indicators and measures must be available for the assessment. There is no source available for such measures. Without target-based indicators and measures, DCOAD can not assess the impact that TST operations have on the overall air campaign.

Because of the above reasons, DCOAD's BAE program manager in consult with the government program manager, Mr. Joe Caroli, decided to re-prioritize the Phase III activities. The change in priorities and level of effort emphasized enhancements to Phase I and II visualization capabilities. In addition, based on feedback from the Phase I and II demonstrations, the government expressed an interest in assessing the impact of terrain on ISR asset probability of detection (Pd) calculations. As a result, this capability was established as a Phase III requirement and successfully implemented. Finally, the government expressed an interest in developing a more realistic and stressful demonstration scenario in preparation for presentations to the operational community. A more complex Iranian scenario was developed and the appropriate files created to support operator demonstrations.

7. Conclusions

The Dynamic Course Of Action Decision-aid (DCOAD) supports predictive battlespace awareness (PBA) for time sensitive targets (TSTs). DCOAD provides operators with predictions of when and where TSTs are likely to be found, and correlates that information with the location and availability of strike and ISR assets. Predictions generated by DCOAD are in the form of probabilistic maps that indicate the likelihood of TST occurrence in a given geographical area, and the Strike and ISR coverage in those areas. These probabilistic maps are overlaid on a situational display to provide operators information regarding "gaps" in TST coverage.

Furthermore, DCOAD generates a composite overlay that provides a single depiction of potential "hot spots" where Strike and ISR coverage is insufficient for anticipated TST activity. Animation is used to show how the situation evolves over time. With this information, operators can develop proactive strategies to find and prosecute TST, as well as plan missions that account for potential TST "pop-ups".

DCOAD combines information from Intelligence Preparation of the Battlespace (IPB), digital terrain data, the ISR collection plan, the Air Battle Plan (ABP), and near-real-time track data to generate the probabilistic overlays. A modular, "plug-in", architecture allows DCOAD potentially be extended to other applications for estimate visualization and for rapid development of estimation algorithms. Estimators provide real-valued estimates of a probability distribution over space and time. When displayed, these estimates are discretized and rendered using "block shapes" at a granularity, in space and time, that the operator specifies.

In the Air Operations Center, DCOAD provides both the Master Air Attack Plan Chief and the ISR Collection Manager an analysis tool to evaluate the daily air battle plan and ensure the appropriate ISR and Strike assets are available over time to counter anticipated TSTs. Additional functionality could be developed to give DCOAD a "drag and drop" capability so planners could reposition available asset locations and routes to ensure potential TST are covered over the active time period of the Air Tasking Order.

The extensible nature of the DCOAD estimator framework and the ability of the OpenMap mapping toolkit to import digital maps makes DCOAD an ideal candidate to support not only theater-level PBA analysis, but also tactical-level PBA. With very little modification, DCOAD could be used to support the US Army and US Marine Corps "close in fight". DCOAD's ability to display and potentially coordinate ISR asset coverage at the street level for patrol and cordon & search operations could increase the effectiveness and safety of Stability and Support Operations.

8. References

"Dynamic Course of Action Analysis", Michael L. Curry, John Stucklen, and Bill Sexton, Proceedings of SPIE, Volume 5423, pp. 66-76, August 2004

Dynamic Course of Action Decision Aid Software Design Document, Rev. 2, May 20, 2005

TCT Occurrence Estimator Software Design Document, Rev. 1, Nov. 17, 2003

Dynamic TST Occurrence Estimator Software Design Document, Rev. 1, Dec. 15, 2004

ISR Estimator Enhancements Software Design Document, Rev. 0, May 20, 2005

9. List of Abbreviations and Acronyms

Acronym	Description
ARGB	Alpha/Red/Green/Blue
A2IPB	Automated Assistance to Intelligence Preparation of the Battlespace
ABP	Air Battle Plan
AOC	Air Operations Center
AODB	Air Operations Database
AWT	Abstract Windowing Toolkit
BAE-AIT	BAE-Advanced Information Technologies
CADRG	Compressed ARC Digital Raster Graphics
CAT	Causal Assessment Tool
COA	Course of Action
DCOAD	Dynamic Course of Action Decision-Aid
DMED	Digital Mean Elevation Data
DMPI	Desired Mean Point of Impact
DTED	Digital Terrain Elevation Data
EBP	Effects Based Plan
GUI	Graphical User Interface
ICSF	Integrated Command, Control, Communications, Computer, and Intelligence (C4I) System Framework
ISR	Intelligence, Surveillance, and Reconnaissance
ISRM	ISR Manager
JFACC	Joint Forces Air Component Commander
JMEM	Joint Munitions Effectiveness Model
LOC	Line of Communication
LOS	Line of Sight
MVC	Model-View-Controller (Architecture)
NAI	Named Area of Interest
PBA	Predictive Battlespace Awareness
Pd	Probability of Damage (for Strike Assets)
Pd	Probability of Detection (for ISR Assets)
SCL	Standard Conventional Load
SDT	Strategy Development Toolkit
SEI-CMM	Software Engineering Institute-Capability Maturity Model
TAI	Target Area of Interest
TBM	Theater Ballistic Missiles
TBMCS	Theater Battle Management Core Systems
TST	Time Sensitive Target
TMS	Track Management System
XML	Extensible Markup Language

Appendix A - Results of the DCOAD Warfighter Analysis Workshop (WAW)

As the DCOAD program entered its final months, the program team and the AFRL program manager, Mr. Joe Caroli, felt it was time to introduce DCOAD to the operational community. As a result a DCOAD briefing and demonstration were presented to the AFC2ISRC/DO, Col Rudolph, and the C2 Battlelab Commander, Col Struk on 21 Jun 05. Both received DCOAD favorably and, as a result, a joint AFC2ISRC/C2BL warfighter analysis workshop was scheduled for 2-3 November 05.

The purpose of the workshop was to allow real world operators to review the current state of the DCOAD technology, assess warfighter usability, and provide feedback on what could/should be done to DCOAD to bring it to an initial operational capability (IOC) within the Air Operations Center (AOC) Weapons System. Eleven operators evaluated DCOAD from across Air Combat Command to include representatives from ACC A2, A3, A8 and AS; ARC2ISRC IN and DO; 505th TRG, 605th TES and AFRL IFSA. Each participant was briefed on DCOAD capabilities, participated in a general DCOAD capabilities discussion, and received hands-on DCOAD training. Each filled a survey that addressed regarding DCOAD. These included:

- Where in the AOC would DCOAD be most useful? What division, team, cell and duty position would find DCOAD most useful?
- Who would use it?
- How would DCOAD be used?
- What decision would DCOAD be used to support?
- Where could DCOAD be used besides the AOC?
- What specific capabilities/enhancements would be needed for initial operation use? (Including internal DCOAD capabilities as well as potential new interfaces / data exchanges).

Participants found the DCOAD could be used in four of the five major divisions in the AOC to include Strategy, Combat Plans, Combat Ops, and the ISR Divisions. They made a point that DCOAD should not be deployed to the AOC as a stand alone capability but rather an integrated capability with existing applications. Specifically, participants indicated DCOAD should be deployed as an integrated capability with:

- Theater Battle Operations Net-centric Environment [TBONE] [as a functional component of the Master Air Attack Planning Tool Kit (MAAPTK)]
- Web-enabled Execution Management Capability [WEEMC] [as a functional component with the Joint TST Manager and Command and Control Personal Computer (C2PC)]
- Collection Management Mission Applications (CMMA)
- TBONE [as a functional component of Information Warfare Planning Capability (IWPC)]

Participants felt that DCOAD could support several decision processes within the AOC to include:

- Optimum ISR Asset placement
- Optimum Strike Asset placement
- Impact of weather on ISR asset placement

- Strategic Planning
- Weapon Target Pairing
- Near real time adjustments to pre-planned ATO

Besides the AOC, participants felt that DCOAD could be used in other organizations to include:

- Distributed Common Ground Station (DCGS)
- Joint Intelligence Centers (JIC)
- Joint Analysis Centers (JAC)
- Army Tactical Operations Centers (TOC)
- Joint Task Force J2, J3, and J5

Participants identified several specific enhancements necessary to bring DCOAD to IOC. These were categorized in three bins; Interfaces, model enhancements, and GUI enhancements. These are summarized below.

- Interface Enhancements
    - TBONE Air Campaign Data Base (ACDB)
    - C2PC (display)
    - Joint Weather Impact System (JWIS)
    - Automatic Update of both Air Operations Database (AODB) and Modernized Integrated Database (MIDB) data
    - Eliminate import & interface selections and mission file. Use only AODB
    - Global Command and Control System – Joint (GCCS-J)
    - Blue Force Tracker
    - Data Link Automated Reporting System (DLARS)
    - Add Air Battle Plan / ATO filter capability
    - Add interface with K-PASA

- Model Enhancements
    - Assess threat to ISR assets
    - Assess impact/potential loss of ISR target deck when ISR assets are moved (link to RSTA/ISR target deck)
    - Include Weather effects
    - Include non-traditional ISR assets (e.g. sniper pods, etc.)
    - Improve ISR footprint models
    - Improve ISR terrain calculations
    - Develop 3-D estimators
    - User access to change estimator and model parameters
    - Improve real time track updates
    - Calculate only strike a/c with ordnance (pre-strike). Remove from calculation a/c that have expended ordnance
    - Ability for users to generate probability estimators in real time
    - Add SEAD/EW estimator
    - Enhance strike estimator to calculate impact of additional tanker support
    - Include target prioritization
    - Enhanced ATO Fly-out
        - Turn radius

        - Real-time tracks enhancements
    - Manual Weapon Target Pairing Capability
    - Add more than two estimators to the composite view
    - Model ground alert assets

- GUI Enhancements
    - Click, drag and drop map objects (ISR orbits, Strike XINT orbits)
    - 3-D visualization
    - Multiple views (open more than one window)
    - Label overlays (so you know which one you are viewing)
    - Threat displays (in particular threat rings)
    - Display airspaces
    - Modify dialogue box labels
    - Modify icons (make more clear)
    - Enhance cursor visibility

Under the current plan, participant inputs will be collated by AFC2ISRC/DO and returned to participating operators so capabilities can be prioritized. A rough order of magnitude (ROM) cost will be developed by the DCOAD contractor for each capability requested. AFC2ISRC/DO in conjunction with the C2 Battlelab will identify potential funding sources and schedule to integrate DCOAD into the AOC baseline.